

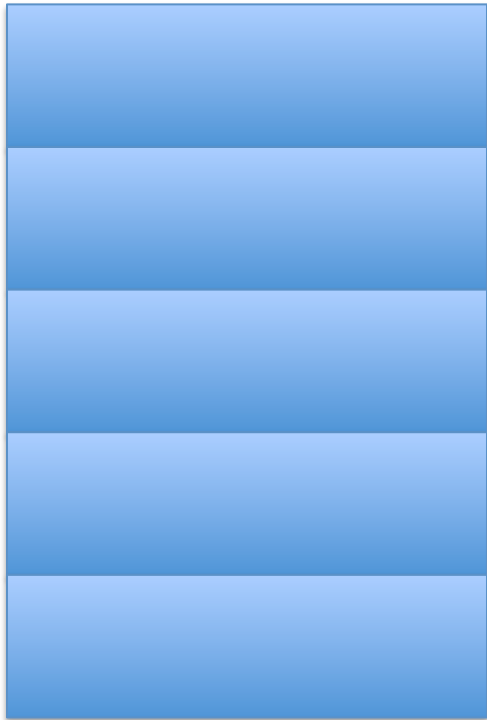
# ***THE COVM***

Peter Wei

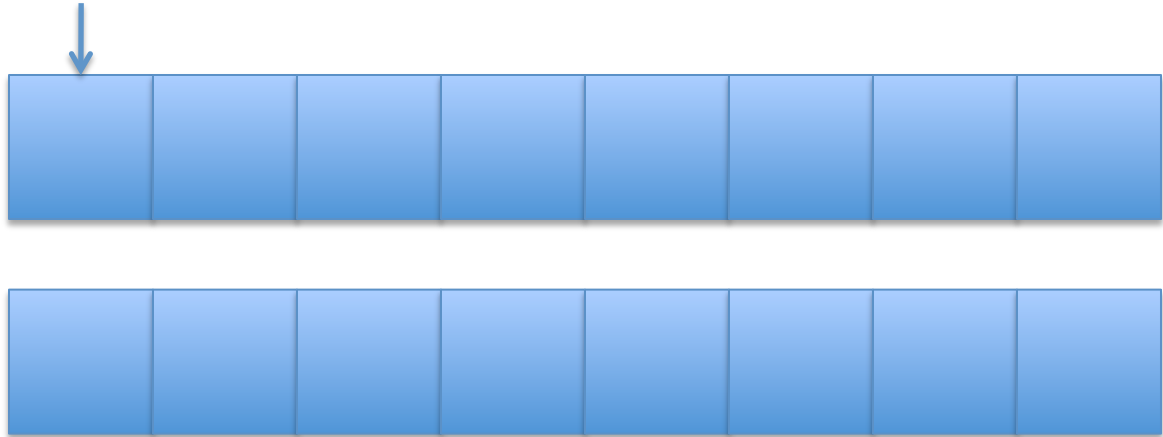
PC (program counter) = 0

stack\_new()

S- stack



ubyte \*P- program pointer



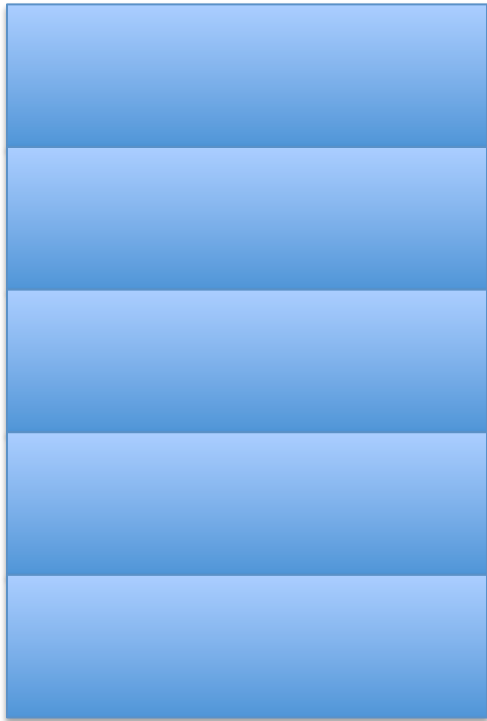
# Example Code

- `int main() {`
- `int x = 5;`      Local variable  
    
- `int y = 2;`      Local variable  
    
- `int z = exp(x, y);`      Local variable  
    
- `return z;`      Not a local variable  
    
- `}`

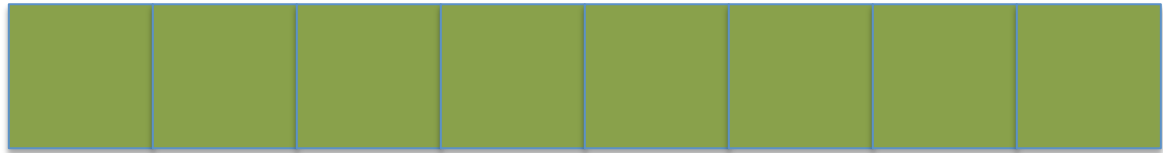
PC (program counter) = 0

stack\_new()

S- stack



ubyte \*P- program pointer



V- variables

xmalloc(sizeof(c0\_value)\*num\_vars)



Call stack



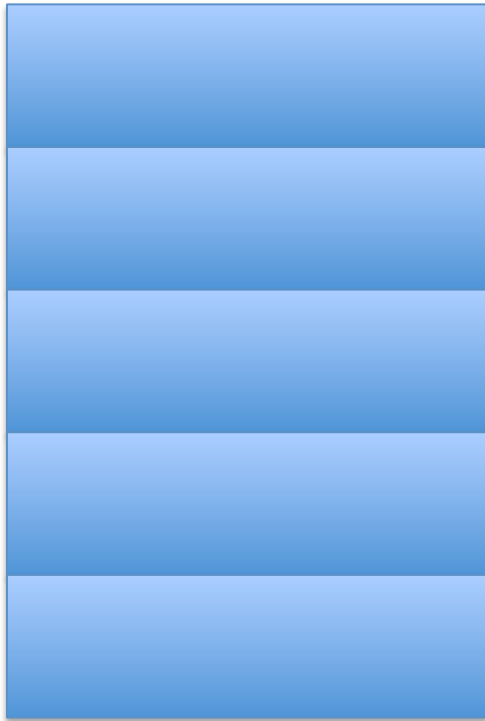
#<main>

```
00 00          # number of arguments = 0
00 03          # number of local variables = 3
00 14          # code length = 20 bytes
10 05  # bipush 5      # 5
36 00  # vstore 0      # x = 5;
10 02  # bipush 2      # 2
36 01  # vstore 1      # y = 2;
15 00  # vload 0       # x
15 01  # vload 1       # y
B8 00 01 # invokestatic 1 # exp(x, y)
36 02  # vstore 2      # z = exp(x, y);
15 02  # vload 2       # z
B0     # return        #
```

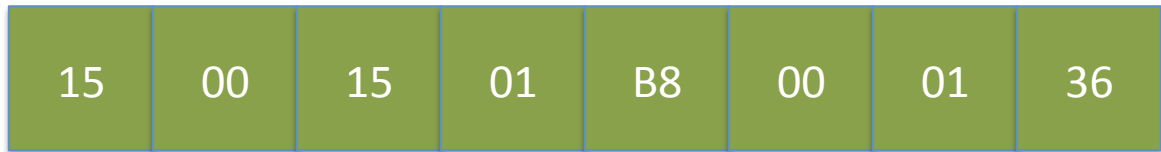
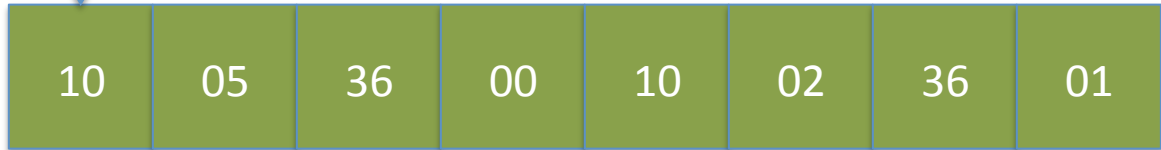
PC (program counter) = 0

stack\_new()

S- stack



P[pc]



V- variables

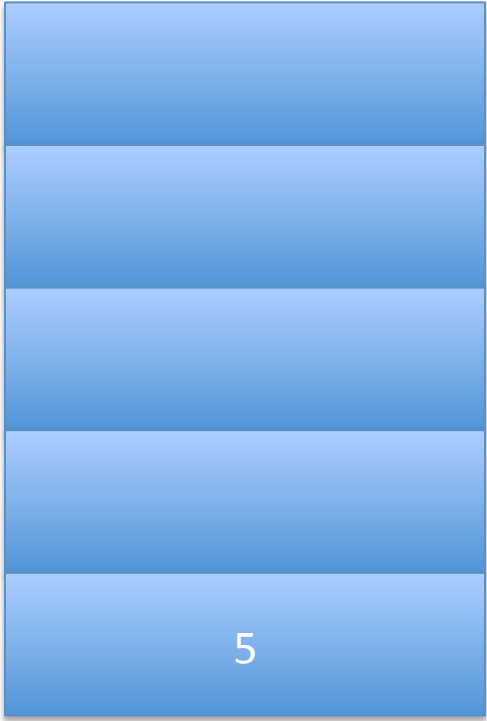


xmalloc(sizeof(c0\_value)\*num\_vars)

Call stack

PC (program counter) = 2

stack\_new()  
S- stack



P[pc]



V- variables

xmalloc(sizeof(c0\_value)\*num\_vars)



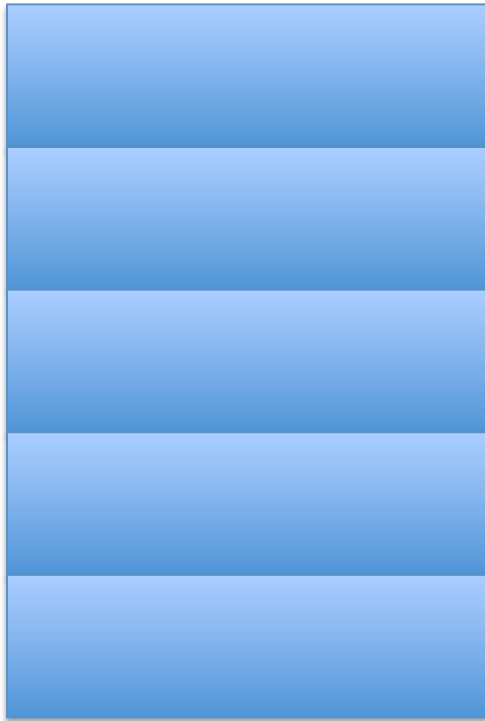
Call stack

After bipush 5

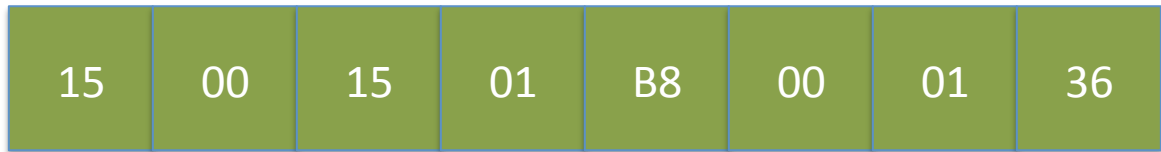
PC (program counter) = 4

stack\_new()

S- stack



P[pc]



V- variables

xmalloc(sizeof(c0\_value)\*num\_vars)

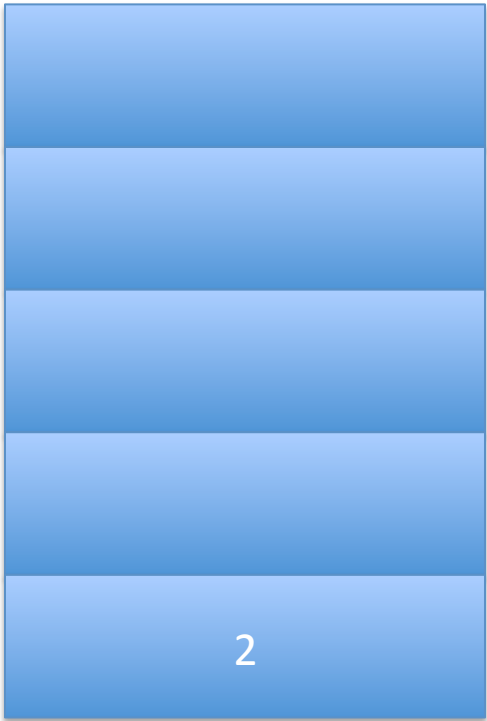


Call stack

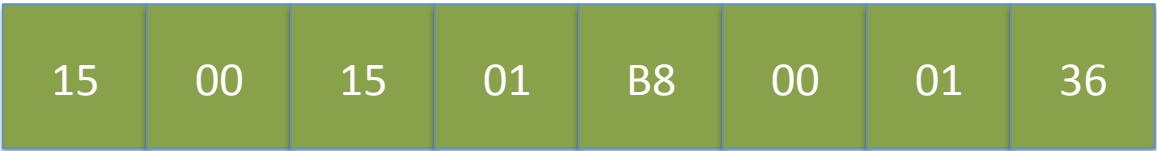
After vstore 0 (pop from stack, put in V[0])

PC (program counter) = 6

stack\_new()  
S- stack



P[pc]



V- variables

xmalloc(sizeof(c0\_value)\*num\_vars)

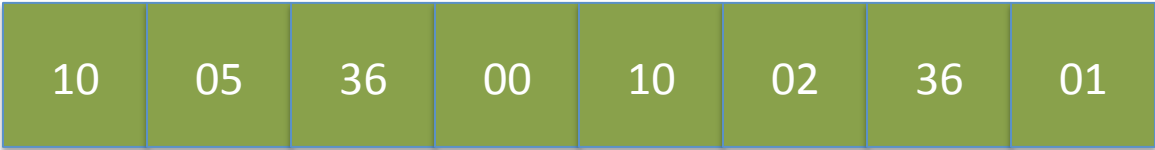
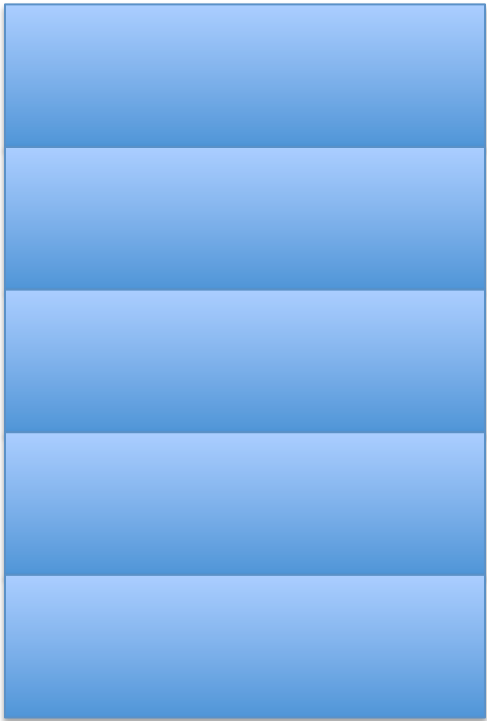


After Bipush 2

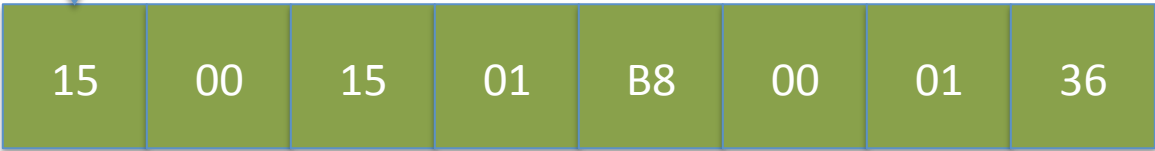
Call stack

PC (program counter) = 8

stack\_new()  
S- stack



P[pc]



V- variables

xmalloc(sizeof(c0\_value)\*num\_vars)



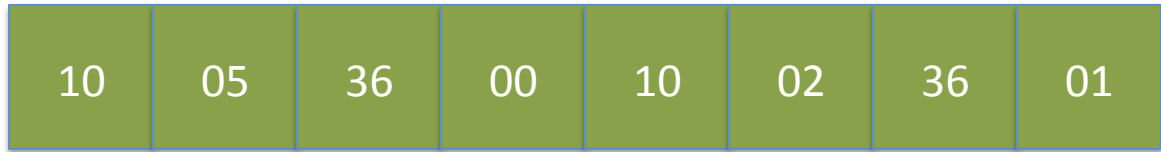
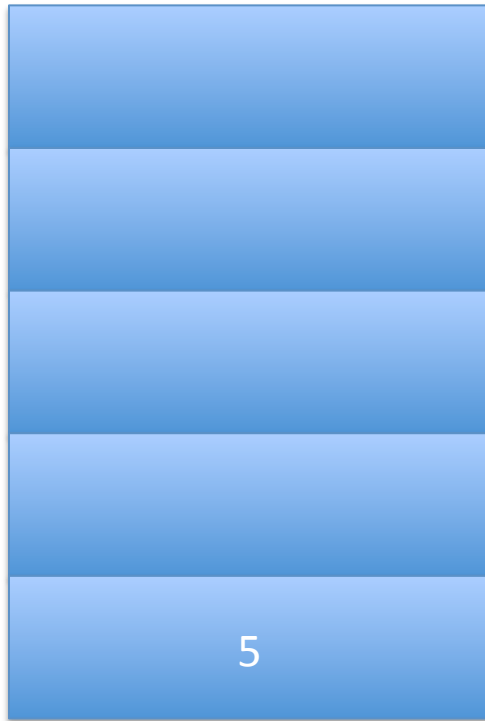
After vstore 1

Call stack

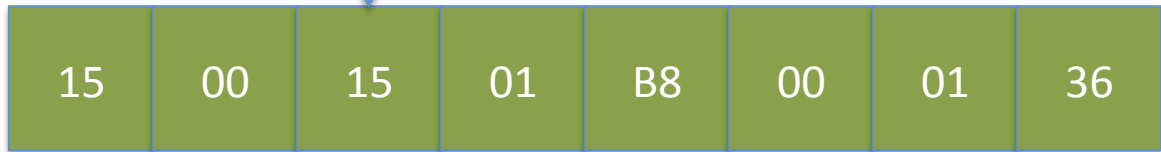
PC (program counter) = 10

stack\_new()

S- stack



P[pc]



V- variables

xmalloc(sizeof(c0\_value)\*num\_vars)



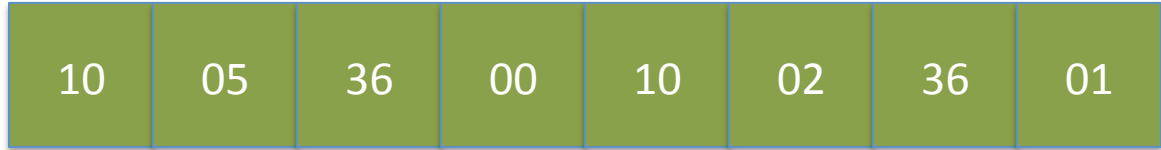
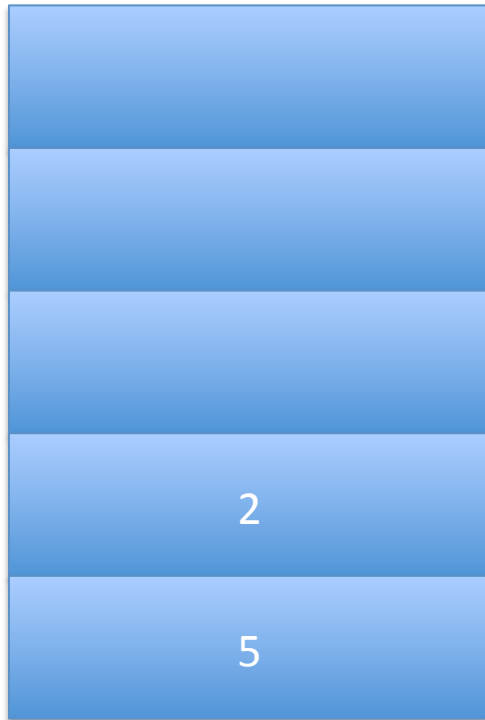
Call stack

After vload 0 (take V[0]  
and push onto S)

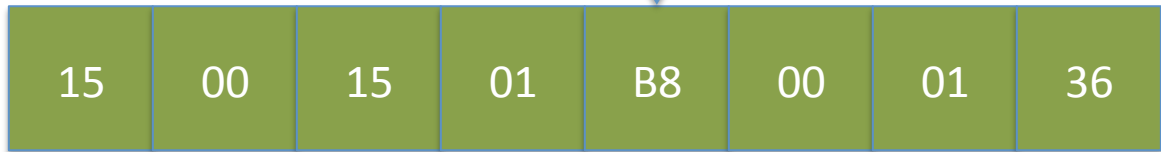
PC (program counter) = 12

stack\_new()

S- stack



P[pc]



V- variables

xmalloc(sizeof(c0\_value)\*num\_vars)



Call stack

After vload 1 (take V[1]  
and push onto S)



# Main Code

```
#<main>
```

```
00 00          # number of arguments = 0
00 03          # number of local variables = 3
00 14          # code length = 20 bytes
10 05  # bipush 5      # 5
36 00  # vstore 0      # x = 5;
10 02  # bipush 2      # 2
36 01  # vstore 1      # y = 2;
15 00  # vload 0       # x
15 01  # vload 1       # y
B8 00 01 # invokestatic 1 # exp(x, y)
36 02  # vstore 2      # z = exp(x, y);
15 02  # vload 2       # z
B0     # return        #
```

Invokestatic- call  
another function, in this  
case, function 1 which is  
exp



```

#<exp>
00 02          # number of arguments = 2
00 02          # number of local variables = 2
00 1E          # code length = 30 bytes
15 01          # vload 1          # e
10 00          # bipush 0         # 0
9F 00 06       # if_cmpeq +6      # if (e == 0) goto <00:then>
A7 00 09       # goto +9         # goto <01:else>
# <00:then>
10 01          # bipush 1         # 1
B0            # return           #
A7 00 11       # goto +17        # goto <02:endif>
# <01:else>
15 00          # vload 0         # b
15 00          # vload 0         # b
15 01          # vload 1         # e
10 01          # bipush 1         # 1
64            # isub             # (e - 1)
B8 00 01       # invokestatic 1   # exp(b, (e - 1))
68            # imul             # (b * exp(b, (e - 1)))
B0            # return           #
# <02:endif>

```

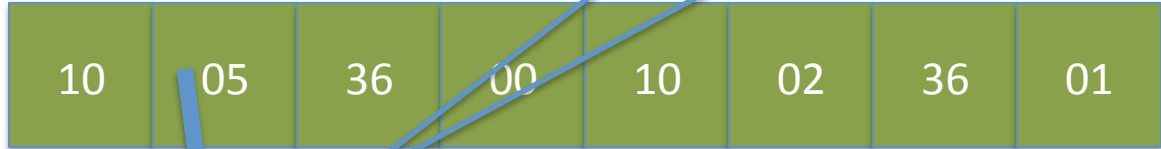
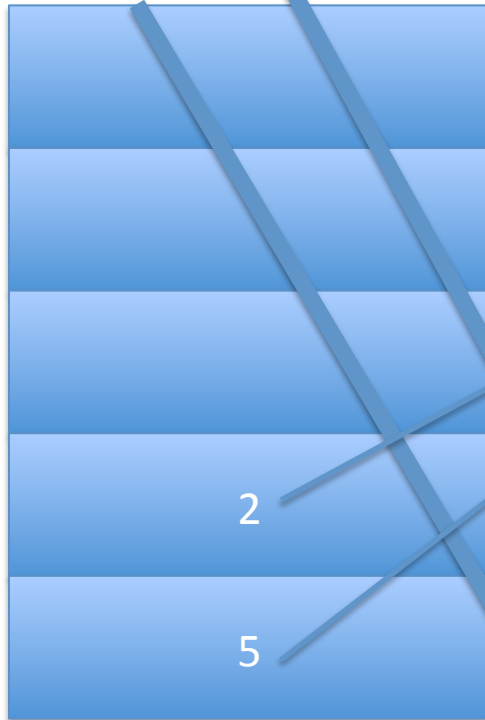
PC (program counter) = 12

new V

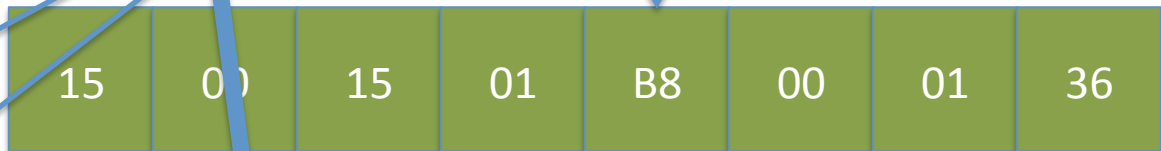


stack\_new()

S- stack



P[pc]

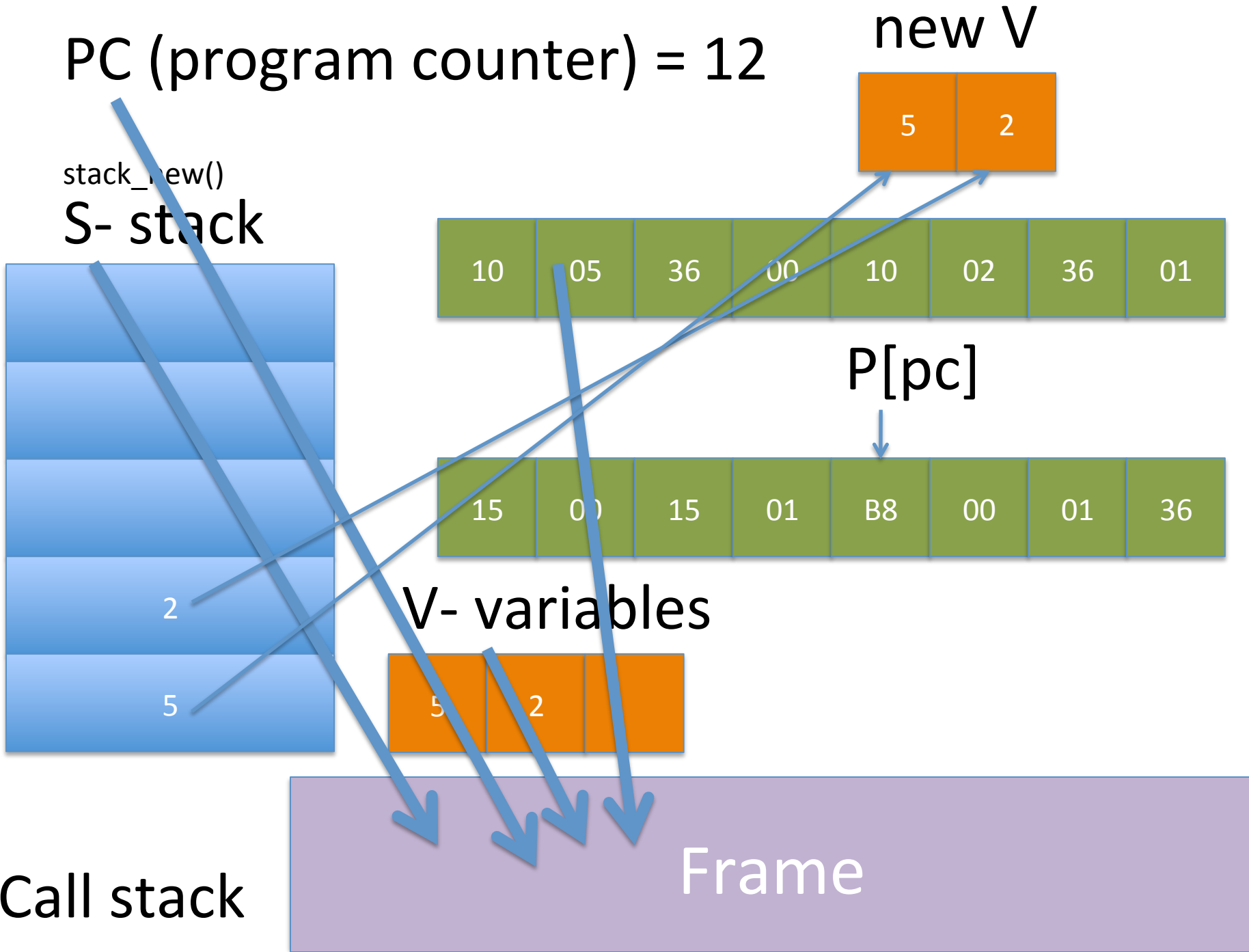


V- variables



Call stack

Frame

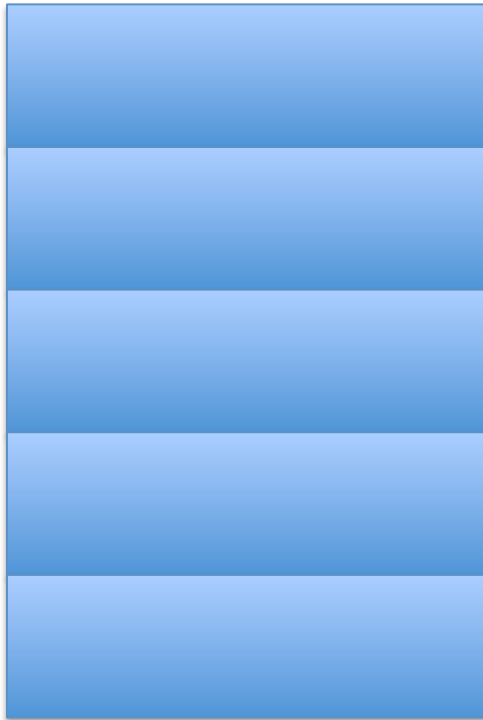


PC reset to 0

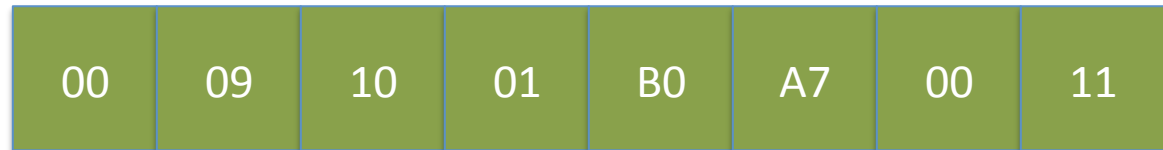
PC (program counter) = 0



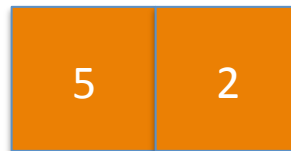
S1



P1[pc]



V1

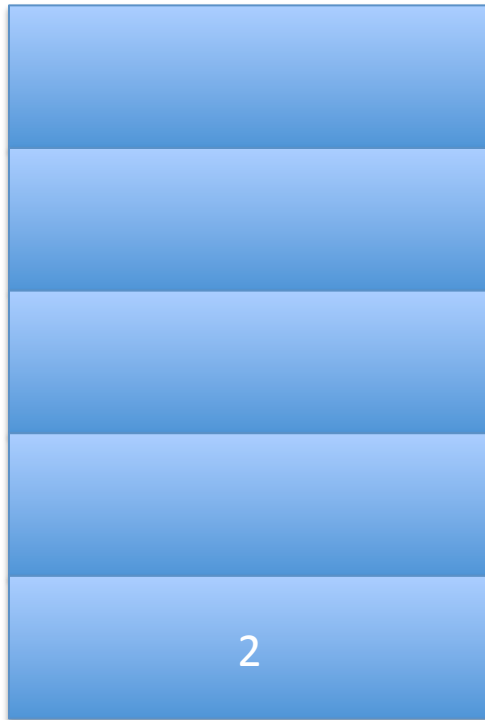


Call stack

S, P, V, pc

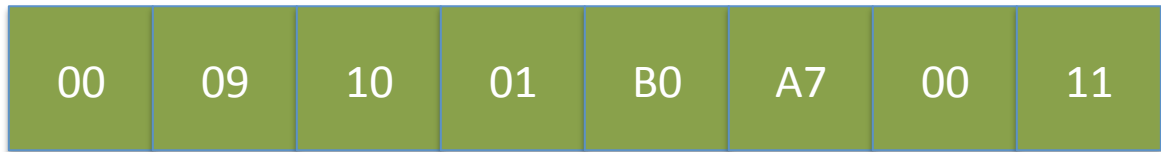
PC (program counter) = 2

S1



Call stack

P1[pc]



V1



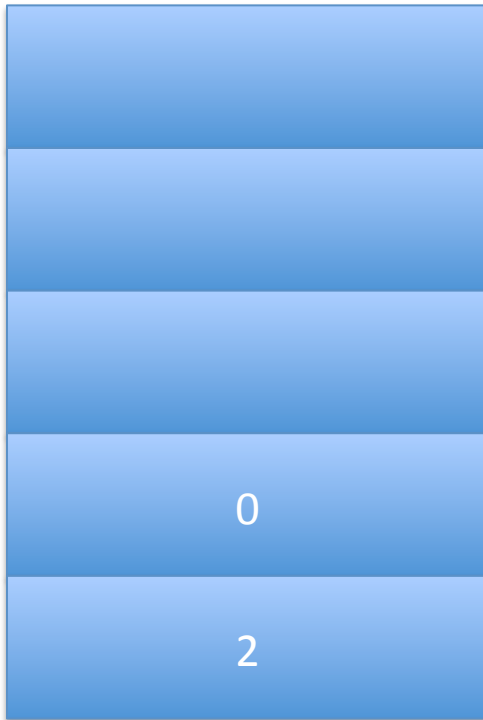
after vload 1

S, P, V, pc

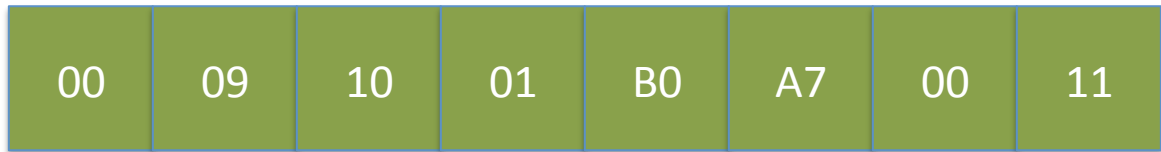


PC (program counter) = 4

S1



P1[pc]



V1



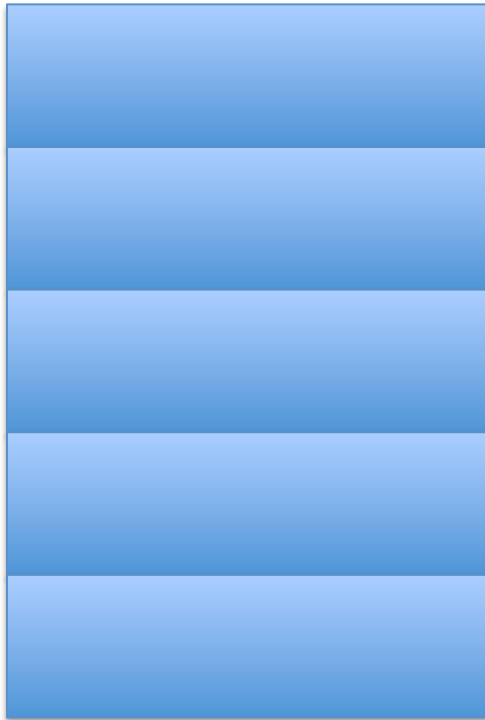
bipush 0

Call stack

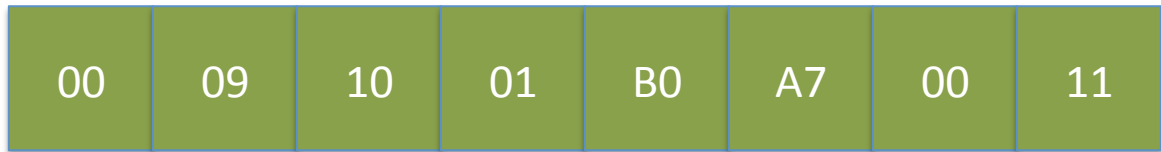
S, P, V, pc

PC (program counter) = 7

S1



P1[pc]



V1



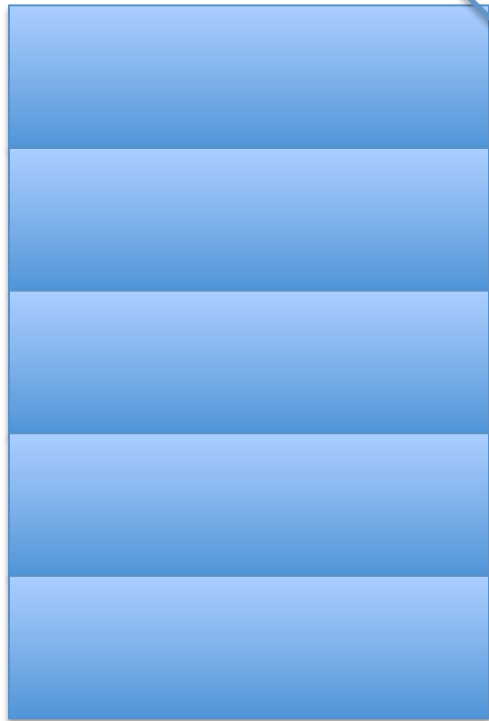
after if\_cmpeq +6-  
pop 2 off S, if =, +6

Call stack

S, P, V, pc

PC (program counter) = 16

S1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1



P1[pc]

after goto +9- jump  
9 bytes

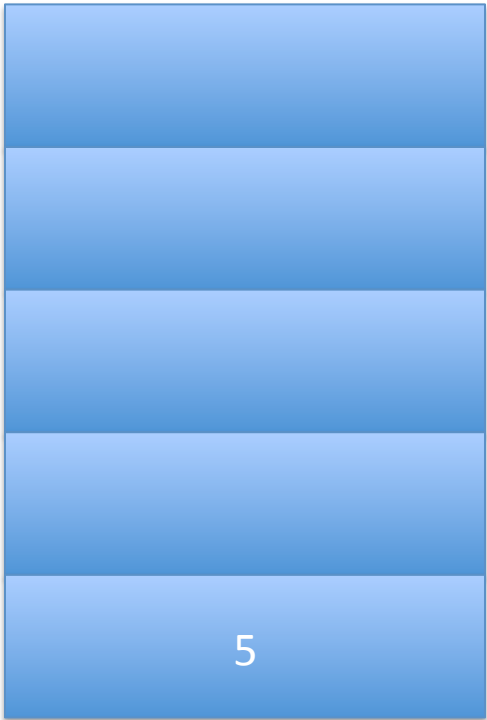
Call stack

S, P, V, pc



PC (program counter) = 18

S1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1



P1[pc]

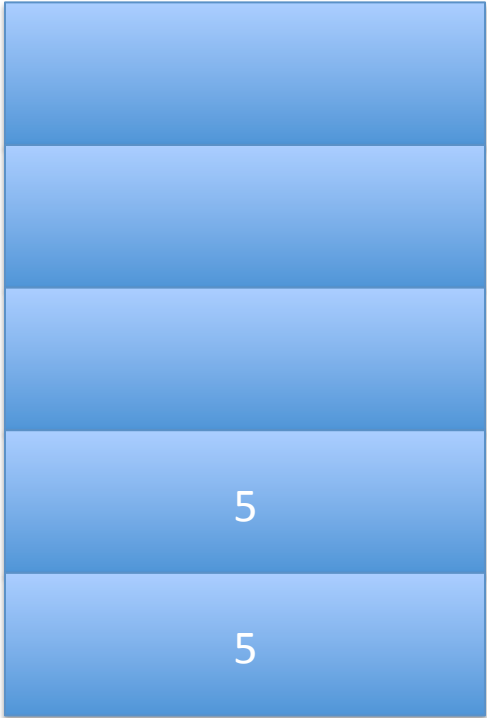
after vload 0

Call stack

S, P, V, pc

PC (program counter) = 20

S1



V1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

P1[pc]

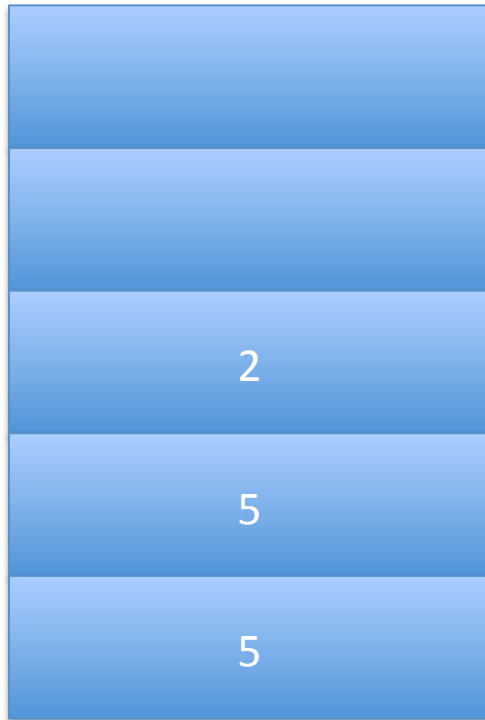
after vload 0

Call stack

S, P, V, pc

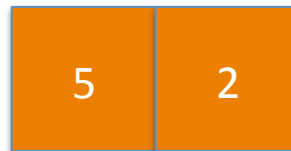
PC (program counter) = 22

S1



Call stack

V1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

P1[pc]

after vload 1

S, P, V, pc

PC (program counter) = 24

S1



Call stack

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1



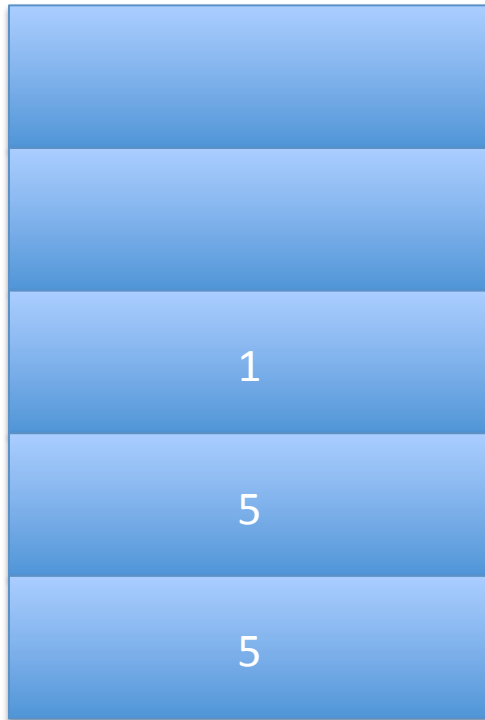
P1[pc]

after bipush 1

S, P, V, pc

PC (program counter) = 25

S1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1



P1[pc]

after isub

Call stack

S, P, V, pc

PC (program counter) = 25

new V

|   |   |
|---|---|
| 5 | 1 |
|---|---|

P1

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B0 | 00 | 01 | 63 | B0 |    |    |

S1

|   |
|---|
|   |
|   |
| 1 |
| 5 |
| 5 |

V1

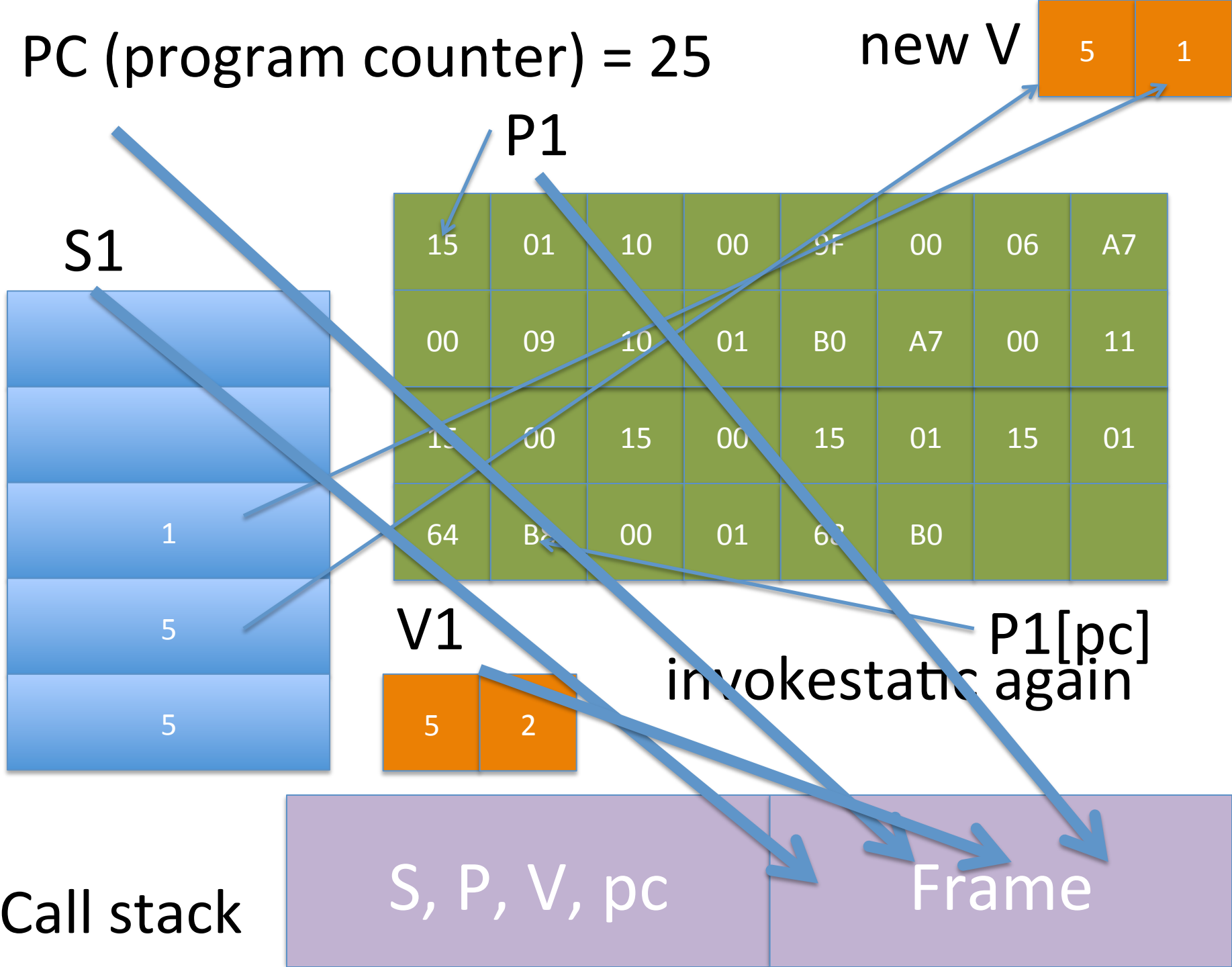
|   |   |
|---|---|
| 5 | 2 |
|---|---|

invokestatic again

Call stack

S, P, V, pc

Frame



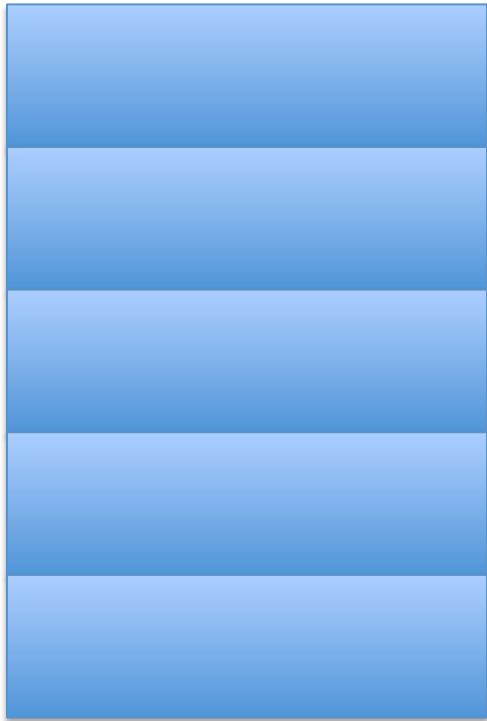
PC (program counter) = 0

P2[pc]



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

S2



V2



invokestatic again

Call stack

S, P, V, pc

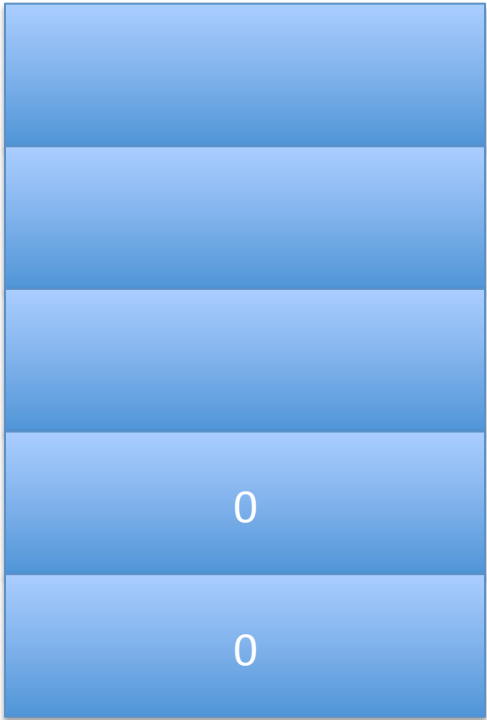
S1, P1, V1, pc1

Skip forward...



PC (program counter) = 4

S3



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

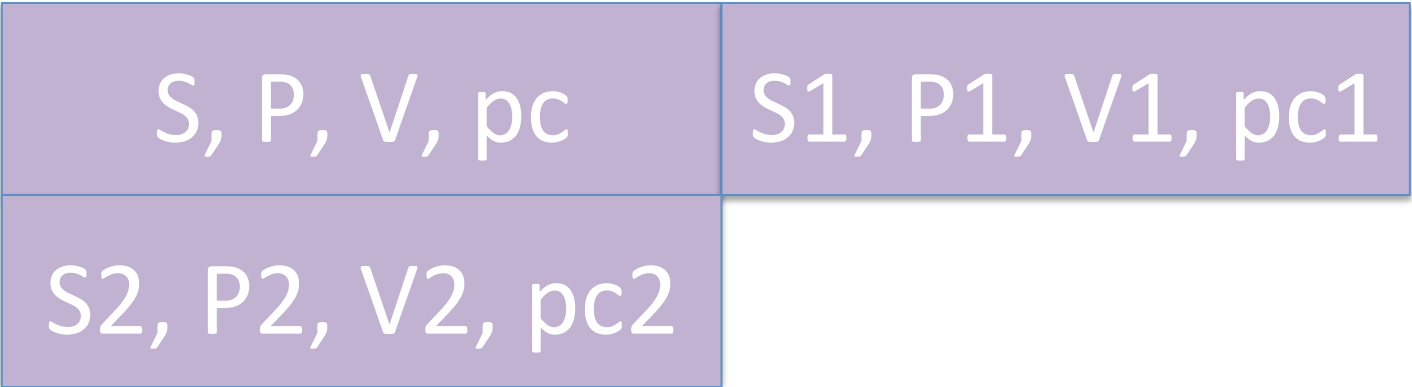
V3



P3[pc]

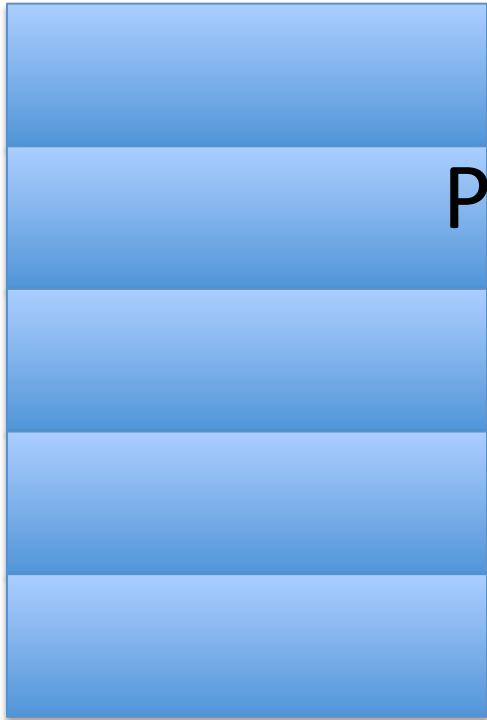


Call stack



PC (program counter) = 10

S3



P3[pc]

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V3

|   |   |
|---|---|
| 5 | 0 |
|---|---|

after if\_cmpeq +6-  
pop 2 off S, if =, +6

S, P, V, pc

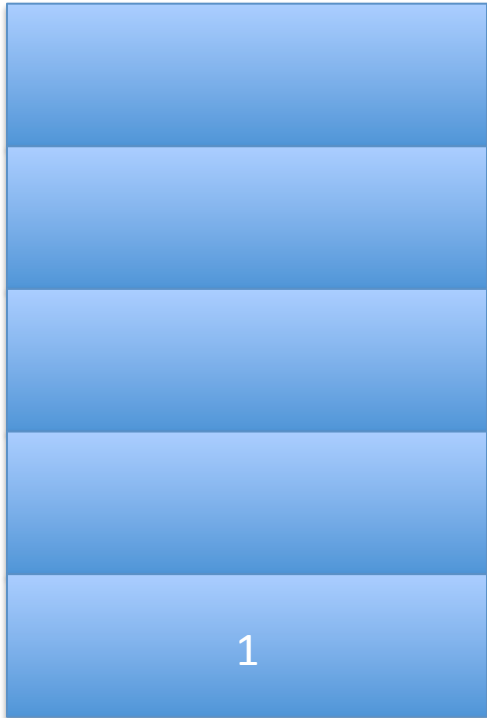
S1, P1, V1, pc1

Call stack

S2, P2, V2, pc2

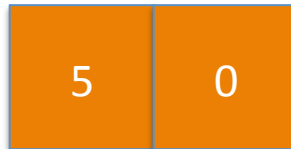
PC (program counter) = 12

S3



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

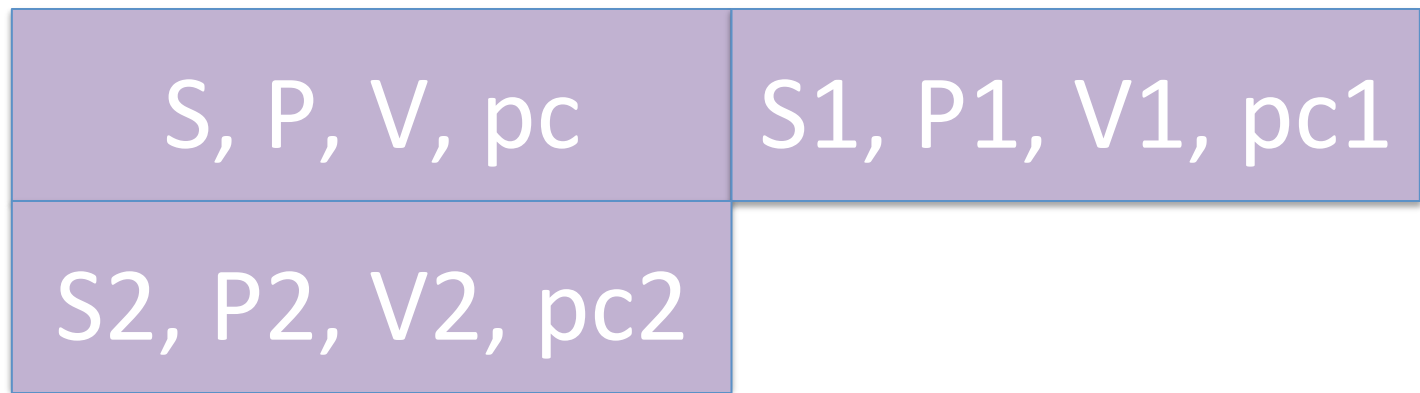
V3



P3[pc]

after bipush 1

Call stack

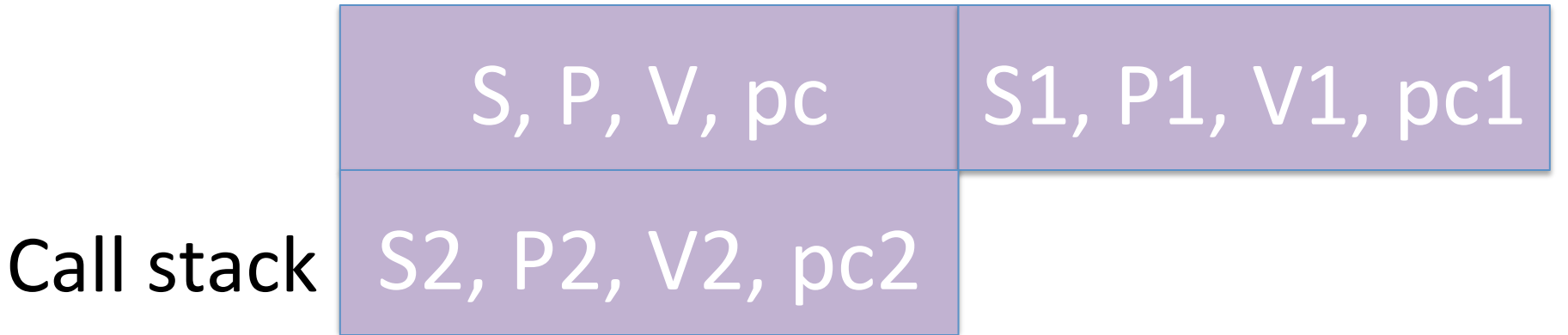
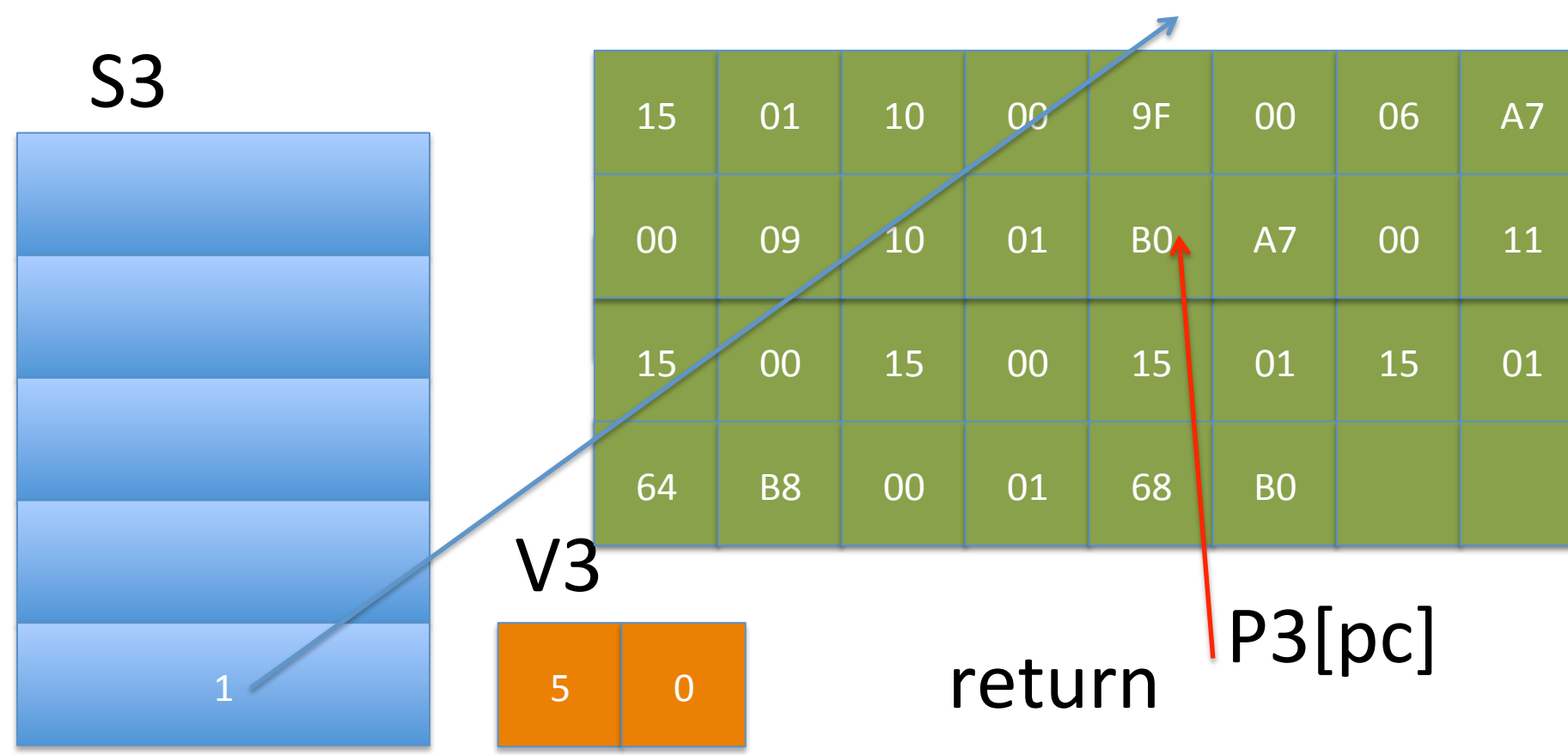


# Now for the magic: return

- 3 things happen in return
- 1) return value is popped off the stack
- 2) the state space is restored using the top stack frame of the call stack ( $S = S_2$ ,  $P = P_2$ ...)
- 3) return value is pushed onto the new state space stack

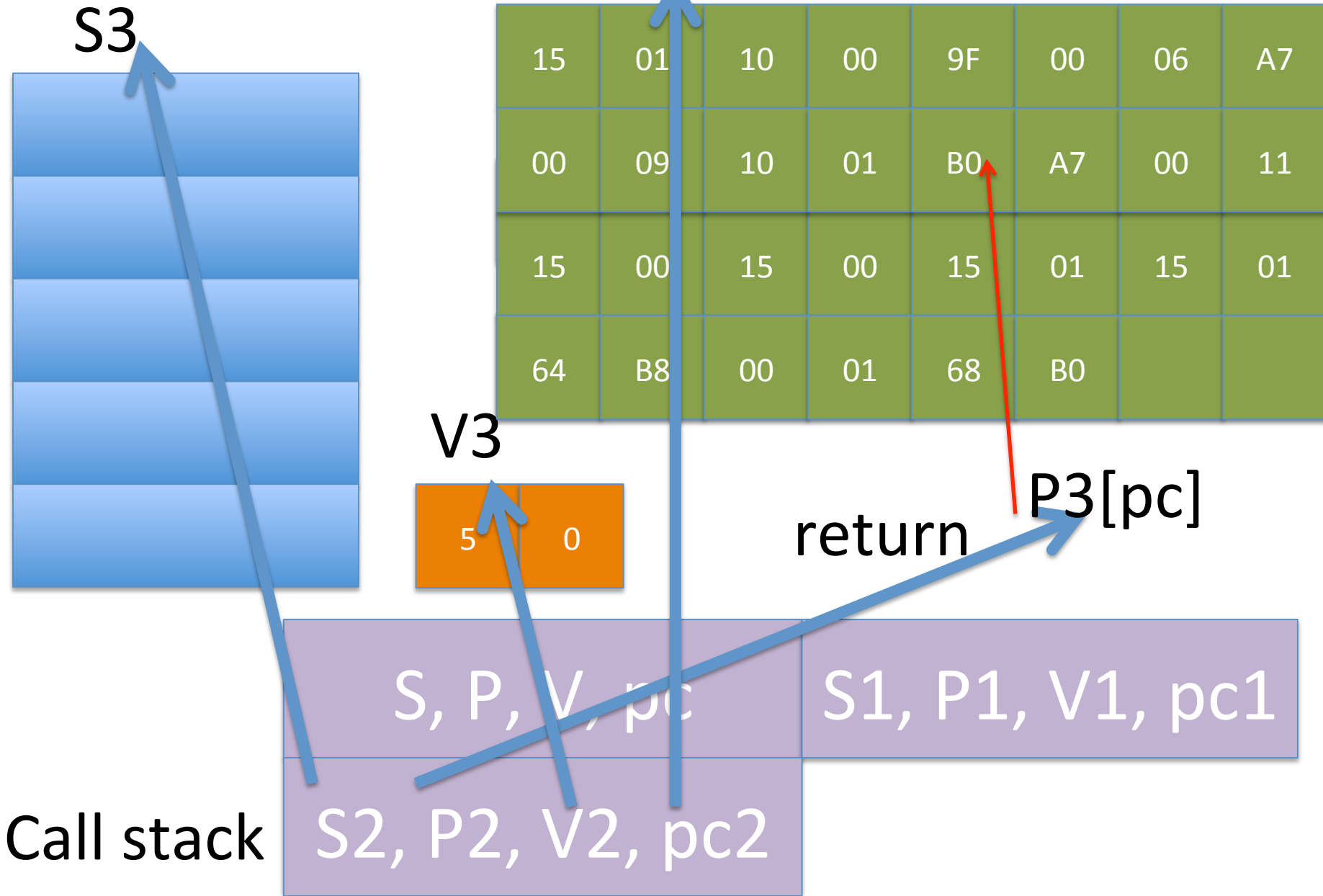
PC (program counter) = 12

hold return value: 1



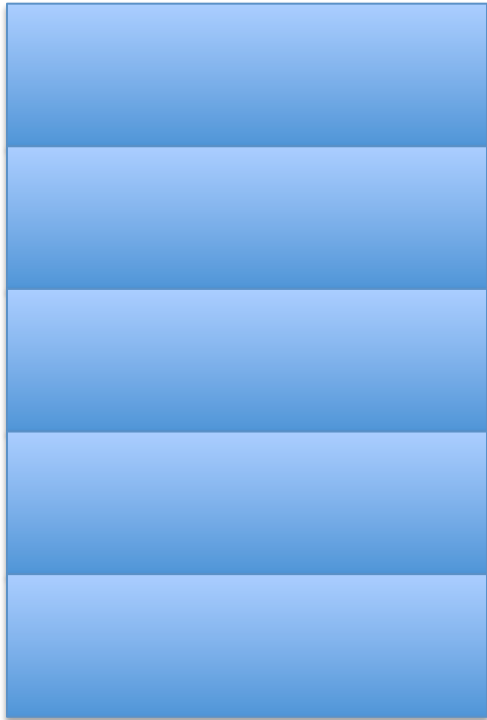
PC (program counter) = 12

hold return value: 1



PC (program counter) = 28      hold return value: 1

S2



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V2

|   |   |
|---|---|
| 5 | 1 |
|---|---|

return      P2[pc]

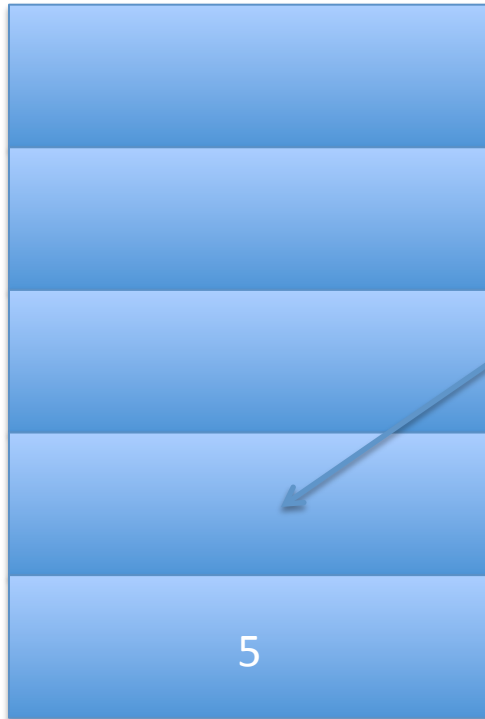
Call stack

|             |                 |
|-------------|-----------------|
| S, P, V, pc | S1, P1, V1, pc1 |
|-------------|-----------------|

PC (program counter) = 28

hold return value: 1

S2



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V2



return P2[pc]

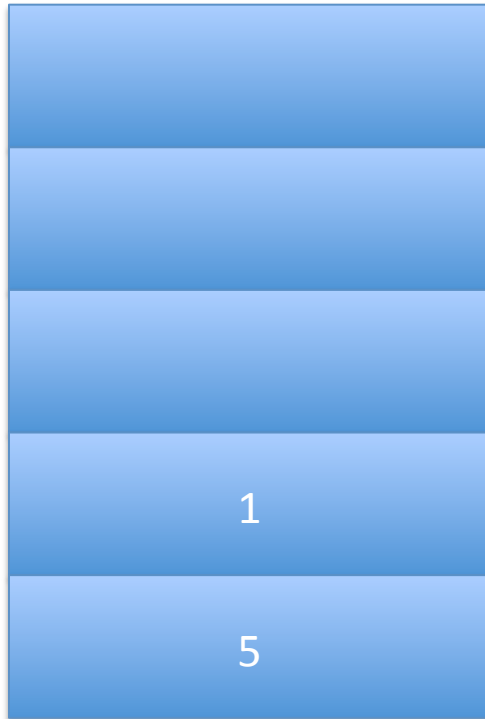
Call stack





PC (program counter) = 28

S2



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V2



return

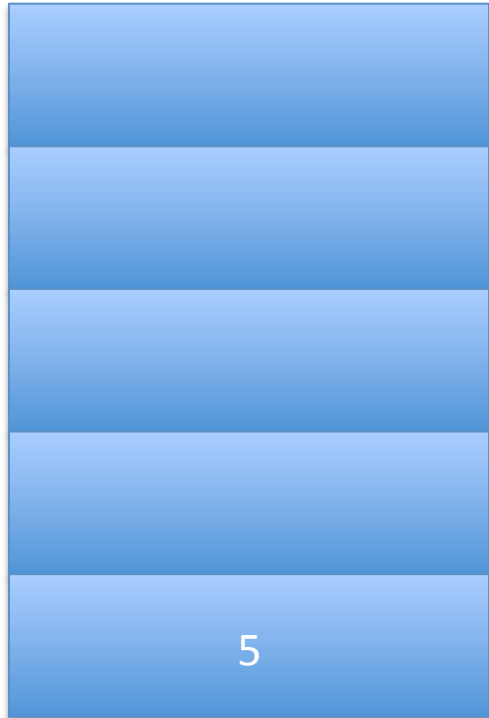
P2[pc]

Call stack



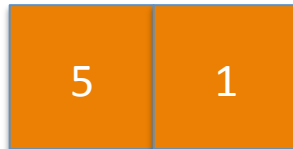
PC (program counter) = 29 hold return value: 1

S2



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V2

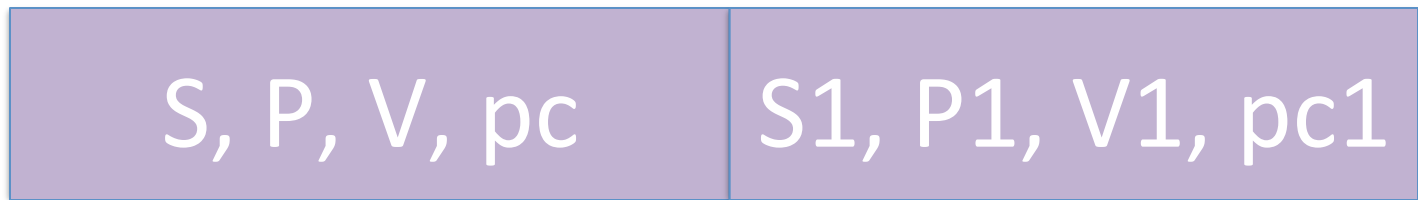


imul

P2[pc]



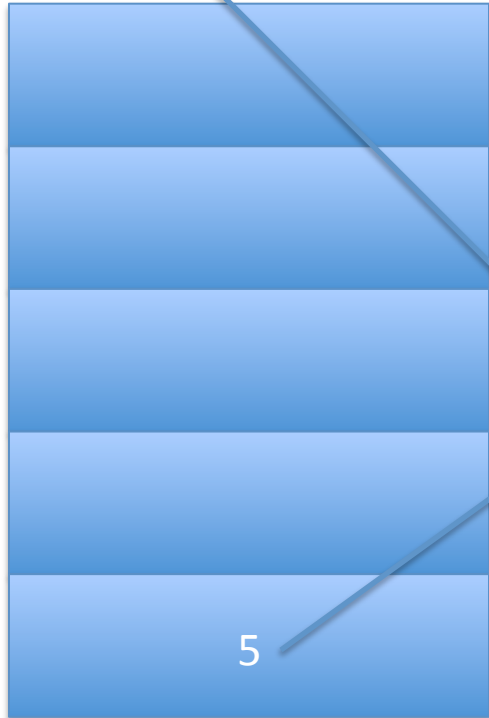
Call stack



PC (program counter) = 29

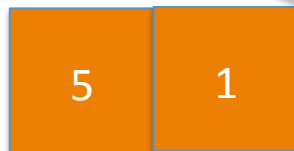
hold return  
value: 5

S2



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V2



return

P2[pc]

Call stack

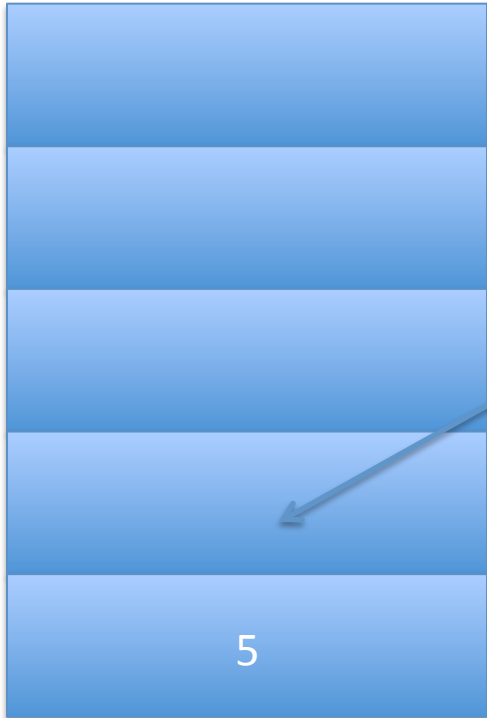
S, P, V, pc

S1, P1, V1, pc1

PC (program counter) = 28

hold return  
value: 5

S1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1



return

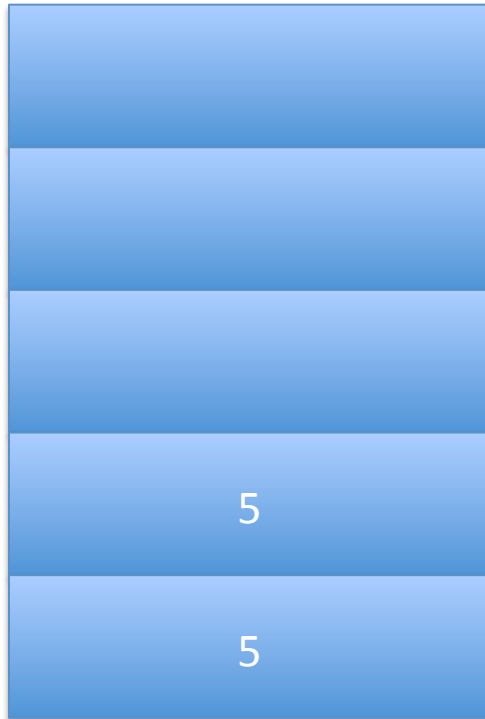
P1[pc]

Call stack

S, P, V, pc

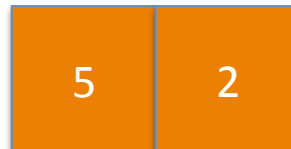
PC (program counter) = 28

S1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1



return

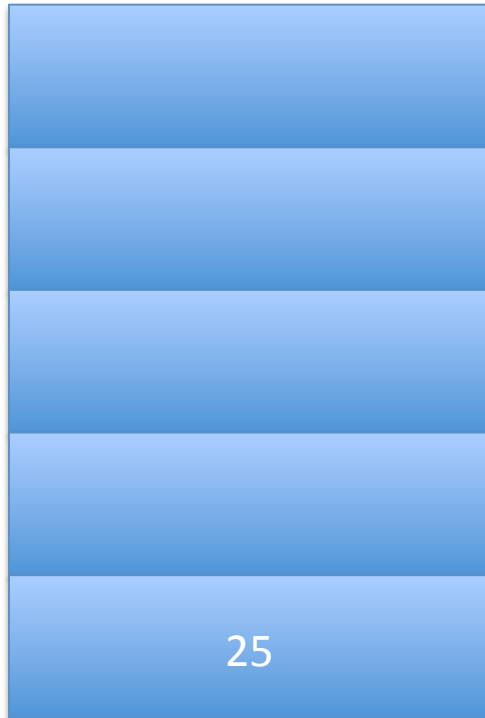
P1[pc]

Call stack

S, P, V, pc

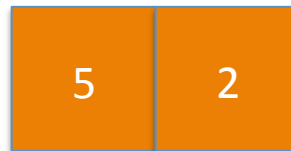
PC (program counter) = 29

S1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1



P1[pc]



imul

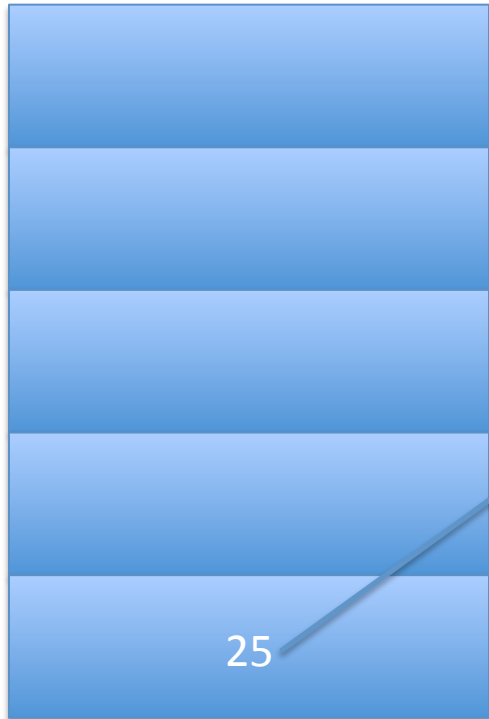
Call stack

S, P, V, pc

PC (program counter) = 29

hold return  
value: 25

S1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1



P1[pc]

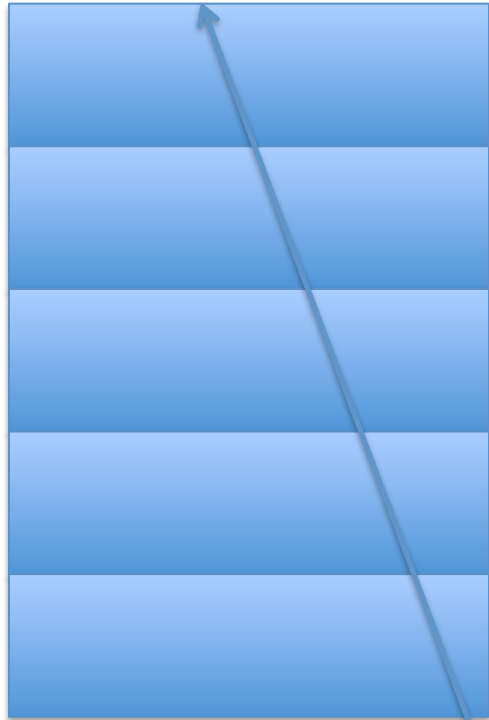
Call stack

S, P, V, pc

PC (program counter) = 29

hold return  
value: 25

S1



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 01 | 10 | 00 | 9F | 00 | 06 | A7 |
| 00 | 09 | 10 | 01 | B0 | A7 | 00 | 11 |
| 15 | 00 | 15 | 00 | 15 | 01 | 15 | 01 |
| 64 | B8 | 00 | 01 | 68 | B0 |    |    |

V1

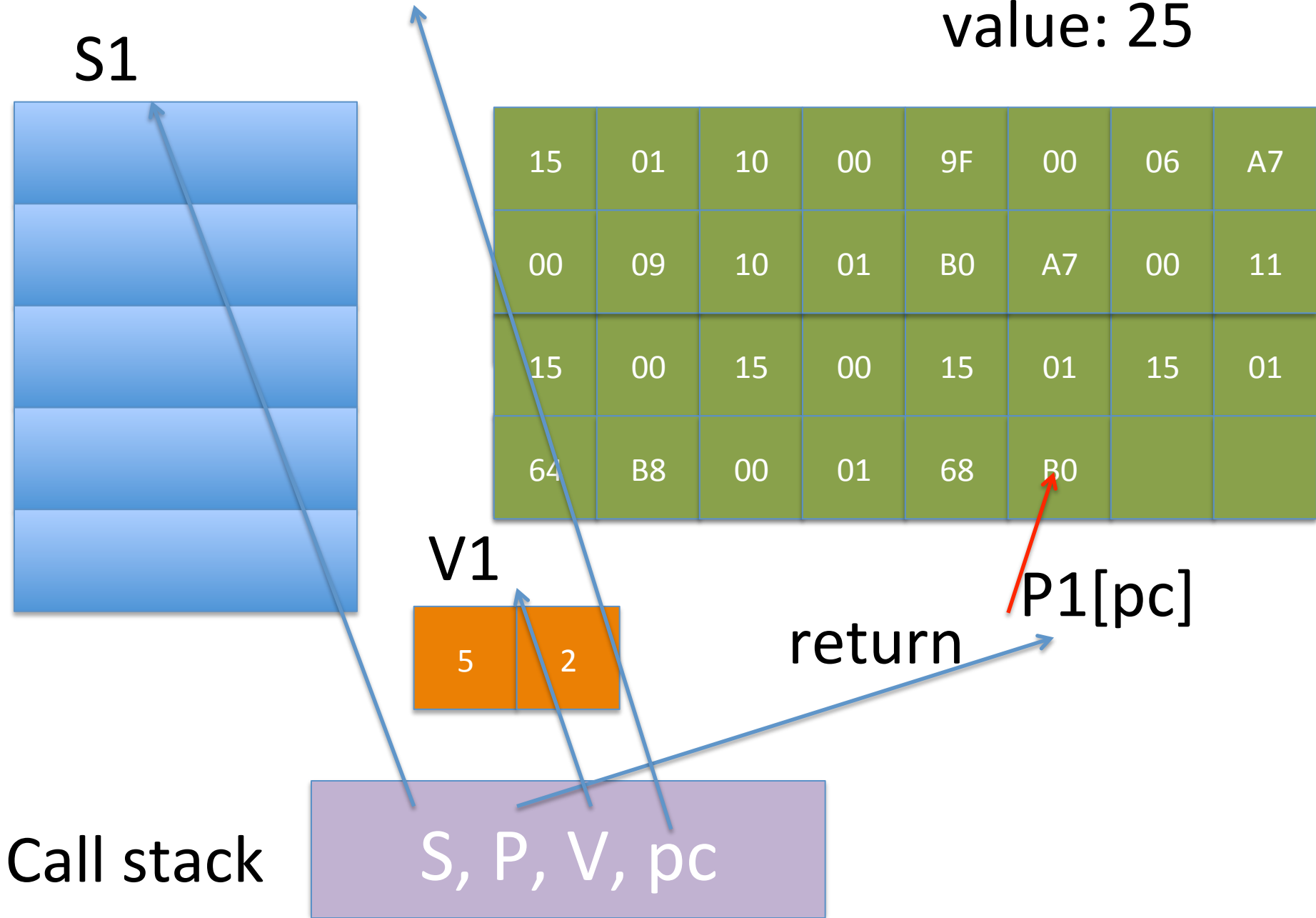


Call stack

S, P, V, pc

return

P1[pc]

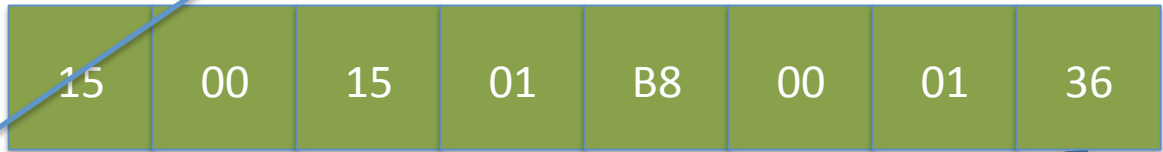
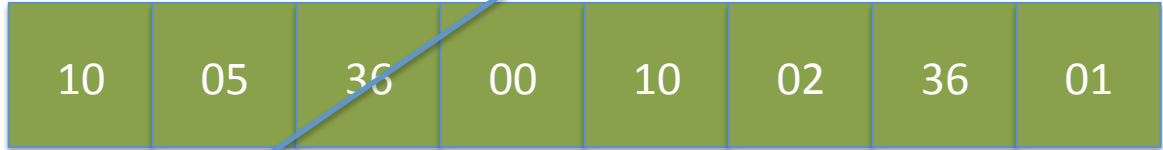
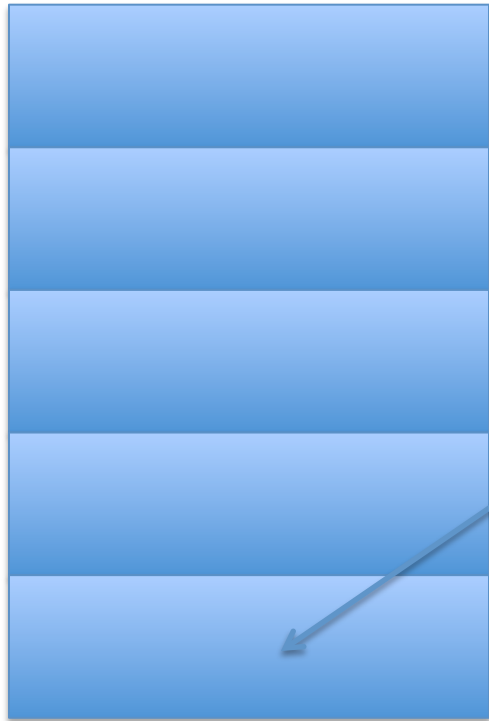




PC (program counter) = 15

hold return  
value: 25

S



V

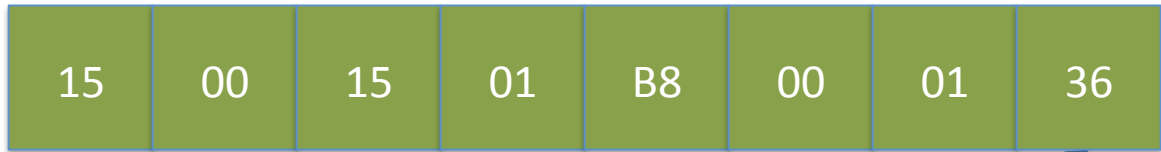
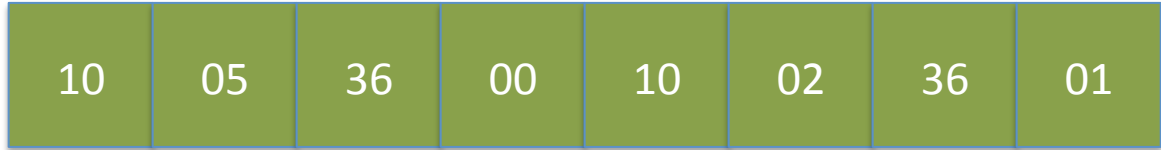
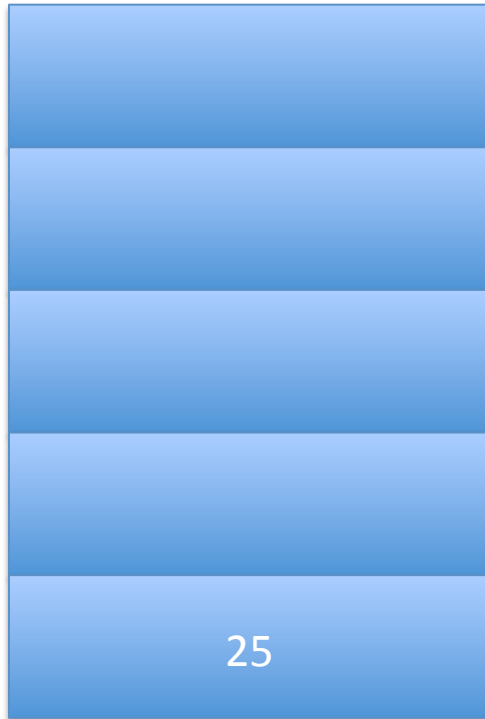


P[pc]

Call stack

PC (program counter) = 15

S



P[pc]

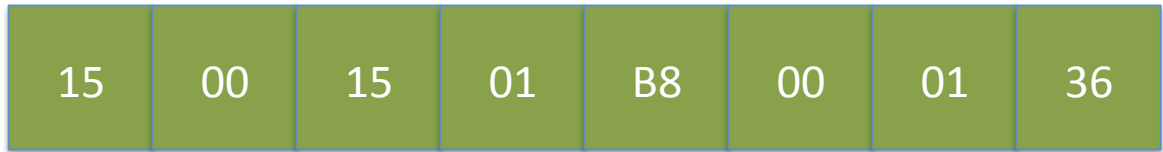
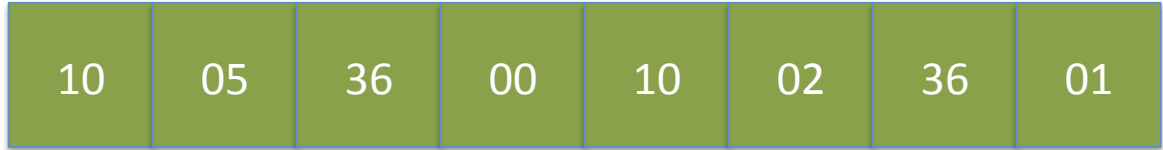
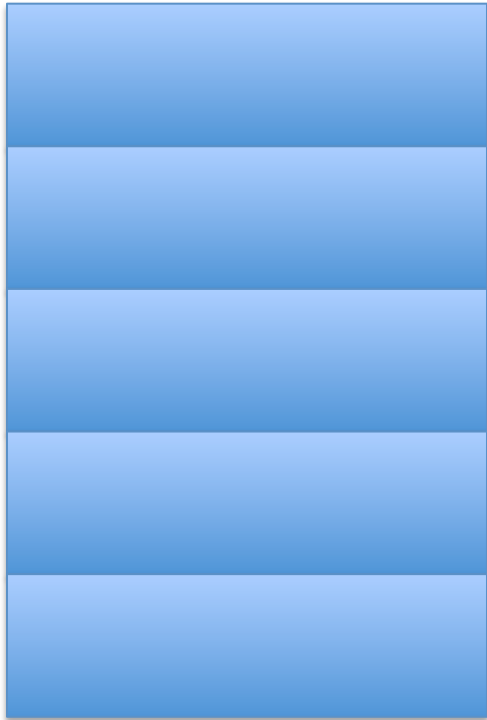
V



Call stack

PC (program counter) = 17

S



V



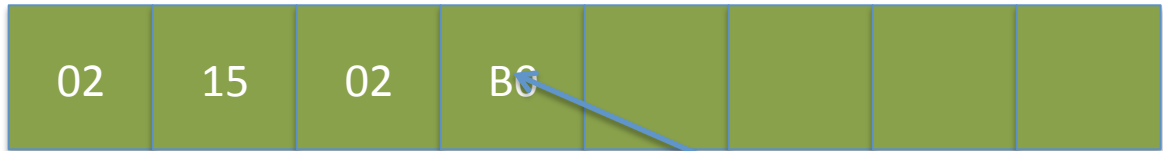
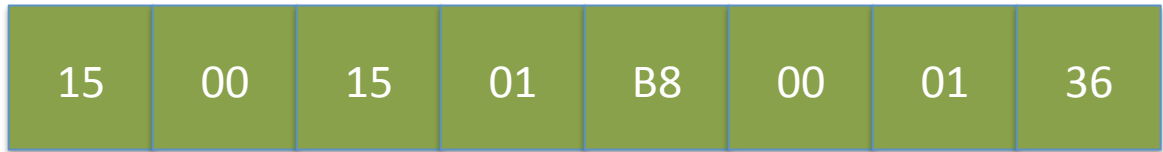
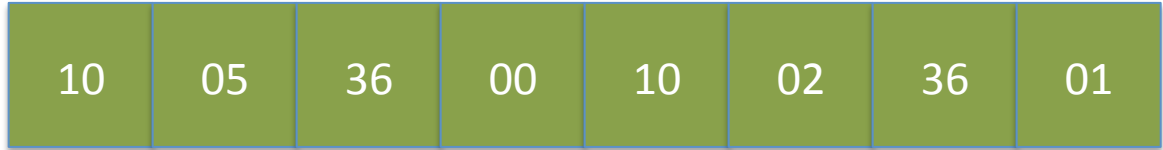
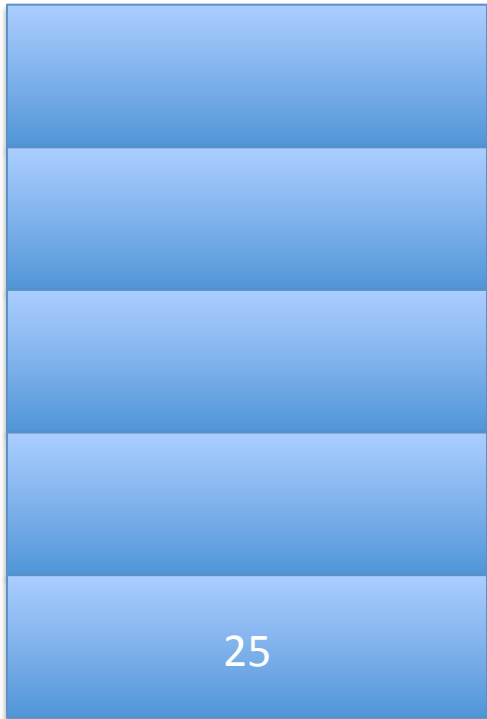
P[pc]

vstore 2

Call stack

PC (program counter) = 19

S



V

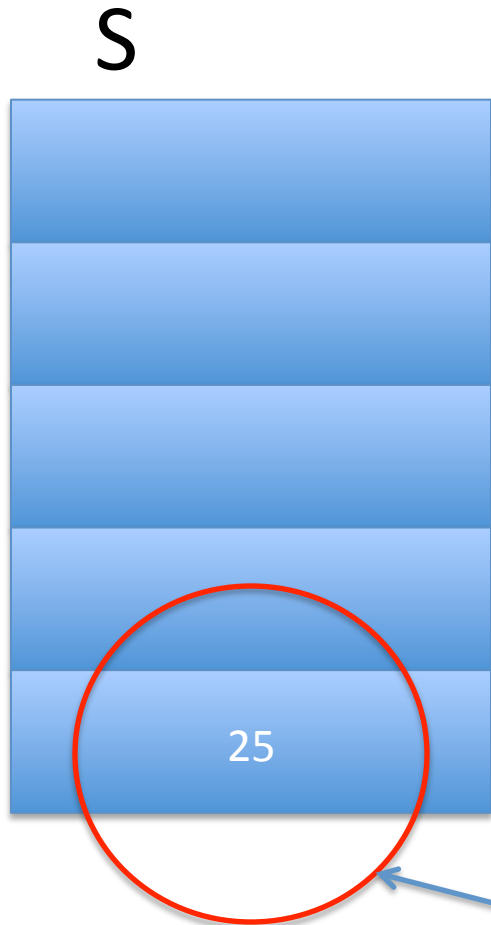


P[pc]

vload 2

Call stack

PC (program counter) = 19



|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 10 | 05 | 36 | 00 | 10 | 02 | 36 | 01 |
|----|----|----|----|----|----|----|----|

|    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| 15 | 00 | 15 | 01 | B8 | 00 | 01 | 36 |
|----|----|----|----|----|----|----|----|

|    |    |    |    |  |  |  |  |
|----|----|----|----|--|--|--|--|
| 02 | 15 | 02 | B8 |  |  |  |  |
|----|----|----|----|--|--|--|--|

V

|   |   |    |
|---|---|----|
| 5 | 2 | 25 |
|---|---|----|

return P[pc]

on return from main, the top of the stack is our final answer

Call stack